TP 2 : Suite du premier

1 Distributeur (Suite)

Continuons notre distributeur afin qu'il puisse servir plusieurs types de boissons.

Remarque : Vous pouvez aussi faire un programme interactif avec la fonction lit entier dans la section programme interactif (un peu plus bas). Par exemple, l'utilisateur peut appuyer sur 1 pour charger le distributeur et 2 pour demander un produit.

1.1 Plusieurs types de produits (Héritage).

1. Nous allons ajouter plusieurs types de produits : les canettes et les friandises. voici comment on crée une classe Canette qui hérite de Produit :

```
class Canette extends Produit{
  /* Les canettes sont tiède a l'origine*/
  private boolean fresh=false;

   Canette(){
   }
   Canette(String nom){
     /*Appelle le constructeur de Produit*/
     super(nom);
   }

  public void refresh(){
    fresh=true;
  }
  public boolean isFresh(){
     return fresh;
  }
}
```

   Une fois cette classe instanciée, elle aura les éléments de la classe produit et les élément que l'on a définie plus haut.

2. l'humain (ici représenté par main) va vérifier la fraîcheur de la canette si ça en est une. Utilisez `produit instanceof Canette` pour vérifier que c'est bien une canette et Caster le Produit en canette si c'est la cas (`canette=(Canette)produit ;`).

3. Faire de même pour friandise.

   Remarque 1 : Pour le chargement vous pouvez faire une canette sur deux produit insérés, utilisez la classe Random de java ou si vous avez fait un menu vous pouvez l'utiliser (cf remarque 4.1).
   Remarque 2 : Il n'y a pas besoin de modifier le distributeur. La classe `Produit` est appelée une classe polymorphe.

1.2 Distributeur réfrigéré

A l'aide des précédentes questions faites la classe `DistributeurRefrigere` qui hérite de la classe `Distributeur` et qui met toutes les canettes dans un compartiment réfrigéré (Pour nous il va simplement appeler refresh() de Canette).

Astuce : Pour appeler la fonction que l'on a redéfinit lors de l'héritage on peut utiliser `super.nomfonction(arg fonction) ;` qui doit être appelé au début de la fonction.

2 Try et Catch ?

2.1 Qu'est ce que c'est ?

Il ne s'agit pas d'un sport mais d'une technique pour récupérer les erreurs. cela se passe comme ceci :

```
class Kamikaze{
  public static void main(String arg[]){
    int a=2;
    int b=0;
    int c;
    try{
      /*serie d'instructions pouvant avoir des erreurs*/
      System.out.println("Essayons ceci");
      c=a/b;
      System.out.println("ça ce passe bien");
    }catch(Exception exception){
      System.out.println("Ooops Il y a un petit problème "+exception);
    }
  }
}
```

1. implémentez ce programme et testez le.
2. Que s'est-il passé pendant l'execution ?
3. Expliquez pourquoi ?

2.2 Et si nous générions des erreurs

```
class Kamikaze2{
  public static void crashTest() throws Exception{
   throw new Exception("Bing Bamm booumm");
  }
  public static void main(String arg[]){
       try{
          System.out.println("lançons le test");
          crashTest();
          System.out.println("Ouf sauvé");
       }catch(Exception exception){
      System.out.println("Ooops Il y a un petit problème "+exception);
    }
  }
}
```

1. implementez ce programme et testez le.
2. Que s'est-il passé pendant l'execution ?
3. Expliquez pourquoi ?

3 Programme interactif

Voici une fonction qui lit un entier dans la console :

```
static int litInt() {
  String s = "";
  int ret=0;
  java.io.BufferedReader b=new java.io.BufferedReader(new java.io.InputStreamReader(System.in));
  try{
    s=b.readLine();
    ret=Integer.parseInt(s);
  }catch(Exception e) {
    System.out.println ("Erreur de lecture");
    System.exit(0) ;
  }
  return ret;
}
```

Ecrire un programme (une suite d'instructions dans une méthode main) qui lit un entier n au clavier, crée un tableau de n entiers, remplit ce tableau avec des entiers lus au clavier puis affiche le plus petit entier du tableau (en parcourant ce tableau). **Question :** Que se passe-t-il lorsqu'on met autre chose qu'un nombre sur la ligne ? Pourquoi ?

4 Static ou pas static ?

Le mot static permet en fait de définir des attributs ou des fonctions propres à la classe. Exemple :

```
class Static{
  static int a=1;
  int b=a;
  void setAB(int x){
    a=b=x;
  }
  public String toString(){
    return "Voici a:"+a+" voici b:"+b;
  }
  public static void setA(int x){
    a=x;
  }
  public static void main(String arg[]){
    Static o1,o2,o3;
    o1=new Static();
    o2=new Static();
    System.out.println("o1 : " + o1);
    System.out.println("o2 : " + o2);
    System.out.println("modifions o1");
```

```
    o1.setAB(3);
    System.out.println("o1 : " + o1);
    System.out.println("o2 : " + o2);
    o3=new Static();
    System.out.println("le petit dernier o3 : " + o3);
    System.out.println("modifions a de la classe");
    Static.setA(9);
    System.out.println("o1 : " + o1);
    System.out.println("o2 : " + o2);
    System.out.println("o3 : " + o3);

  }
}
```

1. Compilez et exécutez la programme ci-dessus.
2. Que remarquez vous ?
3. Expliquez pourquoi.

A quoi sert les mots private et public ?

En fait `private` autorise l'accès de l'objet déclaré qu'aux fonctions de la classe aux quel ces objet appartiennent. `public` autorise l'accès de l'objet déclaré à tout le monde. Par défaut c'est l'autorisation `private package` : c'est à dire qu'il autorise seulement l'accès de l'objet aux fonctions dont les classes sont dans le même package ou le même répertoire.

5 Lecture et écriture sur des fichiers et lecture de la documentation java

5.1 Lecture de documentation

Chercher dans les api la classe `FileReader`.

5.2 Utilisation des acquis

Ecrivez un programme qui prend en argument un nom de fichier et qui affiche le contenue sur la console. Utilisez la fonction `int read() ;` de cette classe et `Character.toString(char )` pour le convertir en Char. Si vous préférez vous pouvez utiliser la classe `BufferedReader` comme dans l'exercice 3 du TP précédent.

5.3 Ecrivons

Faire de même pour l'écriture avec `FileWriter` et `void write(String)`. Vous pouvez utiliser la fonction que l'on a vu en TP1 et enlevez `int Interger.parseInt()`. On peut terminer la saisie grâce à `ctrl+d` dans la console qui va renvoyer la valeur `null`, ou à la vue d'une ligne vide.

6 Pour les plus rapides

1. Vous pouvez reprendre le distributeur et faire la classe Monnayeur qui prend la monnaie et rend l'excèdent. Ainsi en faisant une classe `DistributeurPayant`, vous pouvez intégrer le monnayeur dans le distributeur comme un attribut privé et faire une composition de classes.

Vous n'êtes pas obligé de faire les objets billet et pièce.

2. Si vous avez encore du temps vous pouvez vous amuser à créer la classe `DistributeurMultiproduit`, qui possède une fonction pour sélectionner le numéro d'un produit. Il va de soie que vous devrez changer la façon de ranger les produits afin d'avoir un accès direct.

   La fonction `loadProduit(Produit prodtuit)` peut mettre les produits au bon endroit.

   Je conseille de faire des tableaux par type de produit et un tableau contenant ces derniers. Vous pouvez même ajouter un prix sur chaque produit avec le constructeur adéquat.

3. Vous pouvez aussi utilisez la classe `Exception` pour le retour des erreurs en cas de non paiement ou si il n'y a pas eu de sélection de produit préalablement faite. Par contre vous ne pourrez pas hériter la classe si vous vous contentez d'ajouter un "`throws Exception`" à la fin d'une fonction déjà présente dans la classe mère.

4. Et vraiment, si vous vous ennuyez, vous pouvez faire un fichier avec le listing des produit dans le distributeur.

Références

Pour vous aider, vous pourrez utiliser les deux références ci-dessous. La première contient un tutorial sur le langage et la seconde contient les spécifications de l'API Java (toutes les procédures et les classes déjà définies).

`http://java.sun.com/docs/books/tutorial/`
`http://java.sun.com/j2se/1.5.0/docs/api/`